# ISOSURFACE EXTRACTION

Yarden Livnat
Charles Hansen
Steve Parker
Christopher R. Johnson
*SCI Institute*
*University of Utah*

**Abstract** new advances in isosurface extraction. features in the span space and improvements of view dependent extraction (WISE-¿SAGE)

**Keywords:** Isosurface Extraction, Span Space, View Dependent Extraction

## 1. INTRODUCTION

The need for isosurfaces.

## 2. BACKGROUND

Researchers in many science and engineering fields rely on insight gained from instruments and simulations that produce discrete samplings of three-dimensional scalar fields. Visualization methods allow for more efficient data analysis and can guide researchers to new insights. Isosurface extraction is an important technique for visualizing three-dimensional scalar fields. By exposing contours of constant value, isosurfaces provide a mechanism for understanding the structure of the scalar field. These contours isolate surfaces of interest, focusing attention on important features in the data such as material boundaries and shock waves while suppressing extraneous information. Several disciplines, including medicine [**?**, **?**], computational fluid dynamics (CFD) [**?**, **?**], and molecular dynamics [**?**, **?**], have used this method effectively.

In this chapter, we present an overview of isosurface extraction. We then describe acceleration of isosurface extraction based upon the Space Space. Improvements to Space Space acceleration can be made based upon View Dependent methods and we discuss two such methods.

Let $\varphi : \mathbf{G} \rightarrow \mathbf{V}$ be a given field and let $\mathcal{D}$ be a sample set over $\varphi$, such that,

$$\mathcal{D} = \{d_i\} \qquad d_i \in \mathbf{D} = \mathbf{G} \times \mathbf{V}$$

where $\mathbf{G} \subseteq \mathbf{R}^p$ is a geometric space and $\mathbf{V} \subseteq \mathbf{R}^q$, for some $p, q \in \mathbf{Z}$, is the associated value space. Also, let $d = \|\mathcal{D}\|$ be the size of the data set.

Definition 1 (Isosurface Extraction) *Given a set of samples $\mathcal{D}$ over a field $\varphi : \mathbf{G} \rightarrow \mathbf{V}$, and given a single value $v \in \mathbf{V}$, find,*

$$S = \{g_i\} \quad g_i \in \mathbf{G} \quad \textit{such that,} \quad \varphi(g_i) = v. \tag{1}$$

*Note that $S$, the* isosurface, *need not be topologically simple.*

Approximating an isosurface, $S$, as a global solution to Eq. 1 can be a difficult task because of the sheer size, $d$, of large scientific data sets. Originally, isosurface extraction methods were restricted to structured grid geometry and, as such, early efforts focused on extracting a single isosurface [**?**] from the volumetric data set. Recently, in an effort to speed up isosurface extraction, several methods were developed that could be adapted to the extraction of multiple isosurfaces from structured [**?**, **?**] as well as from unstructured geometry [**?**, **?**]. Nevertheless, for large data sets, these methods do not allow for interactive investigation of the data set, especially for unstructured grids.

Defining $n$ as the number of data cells and $k$ as the number of cells intersecting a given isosurface, most of the existing algorithms have time complexity of $O(n)$. While [**?**] has an improved time complexity of $O(k \log(\frac{n}{k}) + k)$, the algorithm is suitable only for structured hexahedral grids.

## 3. THE SPAN SPACE

Efficient isosurface extraction for unstructured grids is more difficult, as no explicit order, *i.e.*, position and shape, is imposed on the cells, only an implicit one that is difficult to utilize. Methods designed to work in an unstructured domain have to use additional explicit information or revert to a search over the value space, $\mathbf{V}$. The advantage of the latter approach is that one needs only to examine the minimum and maximum values of a cell to determine if an isosurface intersects that cell. Hence, the dimensionality of the problem reduces to two for scalar fields. To facilitate the understanding and analysis of the search over the value space, we introduced [**?**] the notion of the Span Space:

Definition 2 (The Span Space ) *Let $C$ be a given set of cells, define a set of points $P = \{p_i\}$ over $\mathbf{V}^2$ such that,*

$$\forall c_i \in C \qquad associate, \quad p_i = (a_i, b_i)$$

*where,*

$$a_i = \min_j \{v_j\}_i \quad and \quad b_i = \max_j \{v_j\}_i$$

*and $\{v_j\}_i$ are the values of the vertices of cell $i$.*

Using this definition, the isosurface extraction problem can be stated as,

Approach 1 (The Span Search ) *Given a set of cells, $C$, and its associated set of points, $P$, in the span space, and given a value $v \in \mathbf{V}$, find the subset $P_s \subseteq P$, such that*

$$\forall (x_i, y_i) \in P_s \quad x_i < v < y_i$$

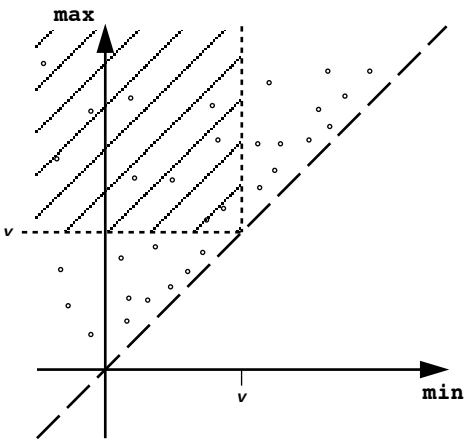


*Figure 1.* Search over the Span Space.

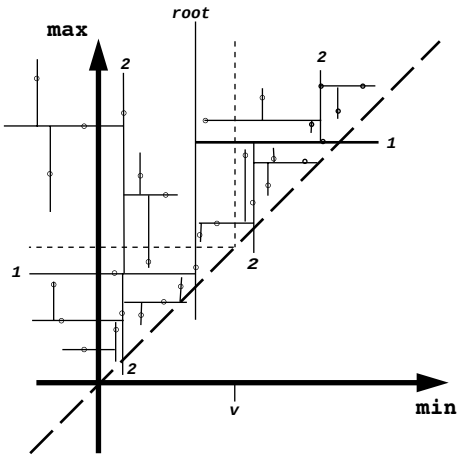


*Figure 2.* Kd Tree decomposition of the Span Space

We note that $\forall (x_i, y_i) \in P_s, \quad x_i \leq y_i$ and thus the associated points will lie on or above the line $y_i = x_i$. A geometric perspective of the span search is given in Fig. 1.

## 3.1. SEARCH OVER THE SPAN SPACE

Given a data set, a kd-tree that contains pointers to the data cells is constructed. Using this kd-tree as an index to the data set, the algorithm can now rapidly answer isosurface queries. Fig. 2 depicts a typical decomposition of a span space by a kd-tree. It is clear that the kd-tree has one node per cell, or span point, and thus the memory requirement of the kd-tree is $O(n)$.

Given an iso-value, $v$, we seek to locate all the points in Fig. 1 that are to the *left* of the vertical line at $v$ and are *above* the horizontal line at $v$. The kd-tree is traversed recursively when the iso-value is compared to the value stored at the current node alternating between the minimum and maximum values at each level. If the node is to the left (above) of the iso-value line, then only the left (right) sub-tree should be traversed. Otherwise, *both* sub-trees should be traversed recursively. For efficiency we define two search routines, *SearchMin* and *SearchMax* which are used on the odd and even levels of the kd-tree respectively, see Figure **??**.

```
SearchMin( node ) {     if (
node.min < isovalue ) {     if
   (node.max > isovalue)
construct a polygon(s) from
node     SearchMax( right child
)    }   SearchMax( left child )
                }
```

```
SearchMax( node ) {     if (
node.min < isovalue ) {     if
   (node.max > isovalue)
construct a polygon(s) from
node     SearchMin( right child
)    }   SearchMin( left child )
                }
```

Estimating the complexity of the query is not straight-forward. Indeed, the analysis of the worst case was developed by Lee and Wong [**?**] only several years after Bentley introduced kd-trees. Clearly, the query time is proportional to the number of nodes visited. Lee and Wong analyzed the worst case by constructing a situation where all the visited nodes are not part of the final result. Their analysis showed that the worst case time complexity is $O(\sqrt{n} + k)$. The average case analysis of a region query is still an open problem, though observations suggest it is much faster than $O(\sqrt{n} + k)$ [**?**, **?**]. In almost all typical applications $k \sim n^{2/3} > \sqrt{n}$, which suggests a complexity of only $O(k)$. On the other hand, the complexity of the isosurface extraction problem is $\Omega(k)$, because it is bound from below by the size of the output. Hence, the NOISE algorithm is optimal, $\theta(k)$, for almost all cases and is near optimal in the general case.

The span space representation has recently been used by Cignoni *et al.* [**?**] to reduce the complexity of the search phase to $O(\log n + k)$ at the expense of higher memory requirements. Shen et al. [**?**] used a

lattice decomposition of the span space for a parallel version on a massive parallel machine.

# 4. VIEW DEPENDENT

The complexity of isosurface extraction algorithms depends on the size of the dataset, $n$, and the size of the isosurface, $k$. The NOISE algorithm achieved a worst case complexity of $O(\sqrt{n}+k)$ while Cignoni *et al.* achieved $O(\log n + k)$. As the dependency on the size of the original dataset was minimized, the size, $k$, of the extracted isosurface became the major factor. Rendering a very large isosurface presents a great challenge even on high-end graphics workstations. An even larger toll has to be paid when remote visualization is involved. To answer this challenge, current research efforts aim to simplify the geometry of the isosurface *after* the isosurface is extracted and *before* it is rendered or transmitted over a network. In effect, the quest is to reduce the complexity of rendering an isosurface to a sublinear complexity with respect to size of the isosurface. However, these methods do not address the effort spent to extract and construct the isosurface in the first place. Furthermore, these methods are slow and have a complexity at least linear with $k$.

## 4.1. BREAKING THE O(K) COMPLEXITY BARRIER

Livnat *et al.* [**?**] proposed to surpass the the $O(k)$ barrier based on the observation that isosurfaces extracted from very large data sets often exhibit high depth complexity for two reasons. First, since the data sets are very large, the projection of individual cells tend to be sub-pixel. This leads to a large number of polygons, possibly non-overlapping, projecting onto individual pixels. Secondly, for some data sets, large sections of an isosurface are internal and thus, are occluded by other sections of the isosurface, as illustrated in Figure 3. These internal sections, common in medical data sets, can not be seen from any direction unless the external isosurface is peeled away or cut off. Therefore, if one can extract just the visible portions of the isosurface, the number of rendered polygons will be reduced resulting in a faster algorithm. Figure 4 depicts a two dimensional scenario. In a view dependent method only the solid lines are extracted whereas in non view dependent isocontouring both solid and dotted are extracted.

The proposed paradigm, which is based on a hierarchical traversal of the data and a marching cubes triangulation, exploits coherency in the object, value, and image spaces, as well as balancing the work be-

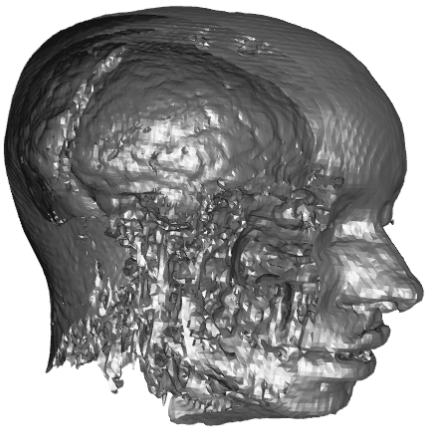

*Figure 3.* A slice through an isosurface reveal the internal sections which can not contribute to the final image.

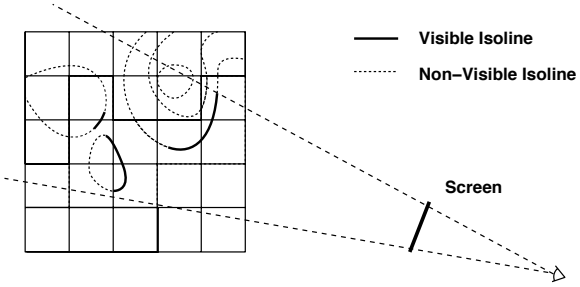


*Figure 4.* A two-dimensional scenario.

tween the hardware and the software. We employ a three step approach, first we augment Wilhelms' and Van Gelder's algorithm [**?**] by traversing down the octree in a front-to-back order in addition to pruning empty sub-trees based on the min-max values stored at the octree nodes. The second step employs coarse software visibility tests for each [meta-] cell which intersect the isosurface. The aim of these tests is to determine whether the [meta-] cell is hidden from the view point by previously extracted sections of the isosurface (thus the requirement for a front-to-back traversal). Finally, the triangulation of the visible cells are forwarded to the graphics accelerator for rendering by the hardware. It is

at this stage that the final and exact [partial-] visibility of the triangles is resolved.

## 4.2. VISIBILITY

Quickly determining whether a meta-cell is hidden and thus can be pruned, is fundamental to this algorithm. This is implemented by creating a virtual screen with one bit per pixel. We then project the triangles, as they are extracted, on to this screen and set those bits which are covered, providing an occlusion mask.

The pruning of the octree nodes is accomplished by projecting the meta-cell on to the virtual screen and checking if any part of it is visible, i.e. if any of the pixels it covers are not set. If the entire projection of the meta-cell is not visible, none of its children can be visible.

We note that it is important to quickly and efficiently classify a cell as visible. A hidden cell, and all of its children, will not be traversed further and thus can justify the time and effort invested in the classification. A visible cell, on the other hand, does not gain any benefit from this test and the cost of the visibility test is added to the total cost of extracting the isosurface. As such, the cell visibility test should not depend heavily on the projected screen area otherwise the cost would prohibit the use of the test for meta-cells at high levels of the octree - exactly those meta-cells that can potentially save the most.

## 4.3. WARPED ISOSURFACE EXTRACTION *(WISE)*

In the WISE algorithm [**?**] we employed hierarchical tiles [**?**] as a mean for fast classification of meta-cells and determining the coverage of extracted triangles. The hierarchical nature of the algorithm ensures that the cost of either of these two operations will not depend highly on their projected area.

A coverage map (a tile) is a rectangular bitmap (we use 8x8) in which each bit represents a pixel in the final image, see Figure 5. The coverage pattern of a convex polygon for a particular tile of the image is computed by combining the coverage maps of the polygon edges.

A key component in the visibility test is the projection of a point, a triangle or a meta-cell onto the screen. Clearly, the cost of performing such transformation for each and every vertex of the projected meta-cells and triangles is too high. In addition, the non-linearity of the perspective transformation prohibits the use of pre-computed transformation table. To accelerate this critical step, we take advantage of the shear-warp factorization [**?**, **?**] of the viewing transformation. The underlying idea

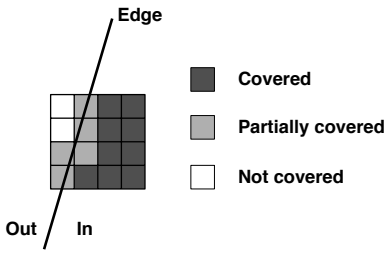


*Figure 5.* An edge tile.

is to factor the viewing transformation into a shear followed by a warp transformation. The data is first projected into a sheared object space that is used to create an intermediate, albeit warped, image. Once this image is complete a warping transformation is applied to create the correct final image.

**4.3.1 Shear But No Warp.** We now note that the visibility on the image plane and on the warped projection plane are the same. In other words, any point in the data set that is visible on the image plane is also visible on the warped projection plane and similarly, points which would be occluded on the image plane are also occluded on the warped plane. It is therefore sufficient to perform the visibility tests on the warped projection plane. The advantage of this approach is two fold. First, the perspective projection is removed. Second, since the shear and scale factors are, with respect to the current view point, constant for each slice we can pre-compute them once for each new view point.

## 4.4. SAGE

The WISE algorithm provides a particular implementation of the view dependent approach. The performance of the WISE algorithm demonstrated the potential benefits of such an approach. The two most prominent weaknesses of the WISE method are the ratio of triangle intersections per screen cell and the fill rate of the screen tiles hierarchy.

In the following we present a new approach to view dependent isosurface extraction that aims at addressing these weaknesses. This approach is based on the WISE method and lessons learned from it and assumes that most of the extracted triangles are fairly small and that the contribution of each triangle to the final image is also small. Based on these assumptions, the triangles are applied to the screen hierarchy in a bottom-up approach (most refined to less refined). The bottom-up

approach helps to restrict the rendering of a triangle to a very small part of the screen hierarchy data structure. The visibility tests of the data meta-cells are still performed in a top-down fashion in order to take advantage of the relative large size of their footprint on the screen.

Another limitation of the WISE algorithm is the restriction to the use of triangles. This restriction is due to the requirement of the tiles method that the projected polygons must be convex. The marching cube algorithm generates between one and four triangles per cell with an average of about 2.05 triangles per cell for the datasets used in this work. Since a cell can be viewed from any direction it is not possible to determine *a priori* if the projection of more than one triangle will be a convex or a concave polygon. The WISE algorithm, thus, projects each and every triangle separately.

Furthermore, as each triangle edge is shared by two triangles, the edge is projected and intersected against the hierarchy twice. The only exceptions are silhouette edges, which are rendered only once. The use of triangles therefore does not permit elimination of those edges that are shared between triangles in the same cell and that form a convex polygon when projected onto the screen.

**4.4.1 A Bottom Up Approach.** To alleviate the problem of projecting many small triangles down the hierarchical tile structure, SAGE employs a bottom-up approach, Figure 6. This approach is based on the observation that the contribution of a small triangle is limited to only a small neighborhood in the hierarchy, *i.e.*, few tiles at the lowest level. This contribution will also be limited in the number of levels in the hierarchy.

The bottom-up approach is realized by projecting the triangles directly on to the bottom level, which is at the screen resolution. Only the tiles that are actually changed by the projection of the triangle will be further checked to see if they cause changes up the hierarchy. Since

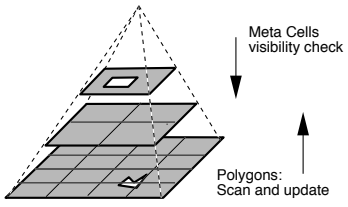


*Figure 6.* Bottom-up and top-down usage in SAGE.

the contribution of the triangle is assumed to be small, its effect up the hierarchy will also be minimal.

**4.4.2 Scan Conversion of Concave Polygons.** One of the disadvantages of a top-down approach based on the hierarchical tiles, is that it is restricted to only convex polygons. In the WISE algorithm, this restriction has forced the projection of triangles only one triangle at a time.

To alleviate this restriction, the SAGE algorithm employs a scan conversion algorithm, which simultaneously projects a collection of triangles and concave polygons. The use of the scan conversion algorithm is made particularly simple in SAGE due to the bottom-up update approach. The projected triangles and polygons are scan-converted at screen resolution at the bottom level of the tile hierarchy before the changes are propagated up the hierarchy. Applying the scan line in a top-down fashion would have made the algorithm unnecessarily complex.

Additional acceleration can be achieved by eliminating redundant edges, projecting each vertex only once per cell and using triangles strips or fans. To achieve these goals, the marching cubes lookup table is first converted into a triangles fans format. The usual marching cubes lookup table contains a list of the triangles (three vertices) per case.

A comparison of the WISE and the SAGE algorithms with respect to the number of polygons and edges that are projected onto the hierarchical tiles is shown in Figure 7.

## 4.5. RENDERING POINTS

Another potential saving is achieved by using points with normals to represent triangles or [meta-] cells which are smaller than a single pixel. This is an improvement over the WISE algorithm as the exact location of the each screen pixel center is known during the scan line and the visibility tests. Whenever a nonempty [meta-] cell is determined to have a size less then a single pixel and its projection covers the center of a pixel, it is represented by a single point. Figure 8 shows an example in which some of the cells are far enough such that they can be rendered as point. On the left is the image as seen by the user while on the right is a close up view of the same extracted geometry (*i.e.*, the user zoomed in but did not extract the geometry based on the new view point).

# 5. SUMMARY

We have described an overview of isosurface extraction. By utilizing the space space, one can rapidly accelerate the search for existing iso-

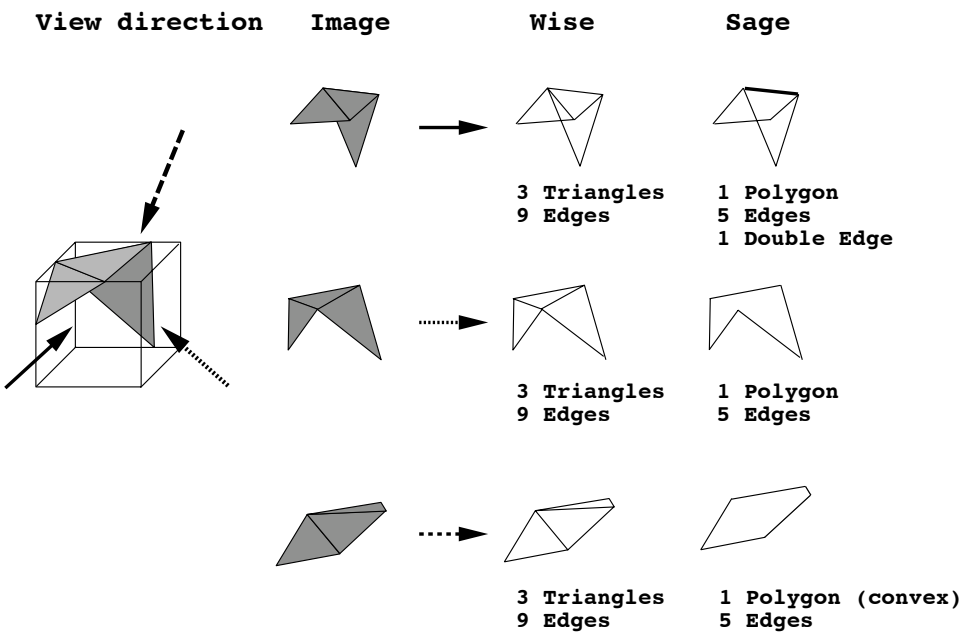


*Figure 7.* Comparison between WISE and SAGE.

surfaces within scientific data sets. When the depth complexity is high, leveraging view dependent methods can be quite effective. We have described two such methods: Warped ISosurface Extraction (WISE) and SAGE.

## Acknowledgments

This work was supported in part by awards from the DOE ASCI, the DOE AVTC, the NIH NCRR, and the NSF. The authors would like to thank Peter-Pike Sloan and Phil Sutton for their contributions.

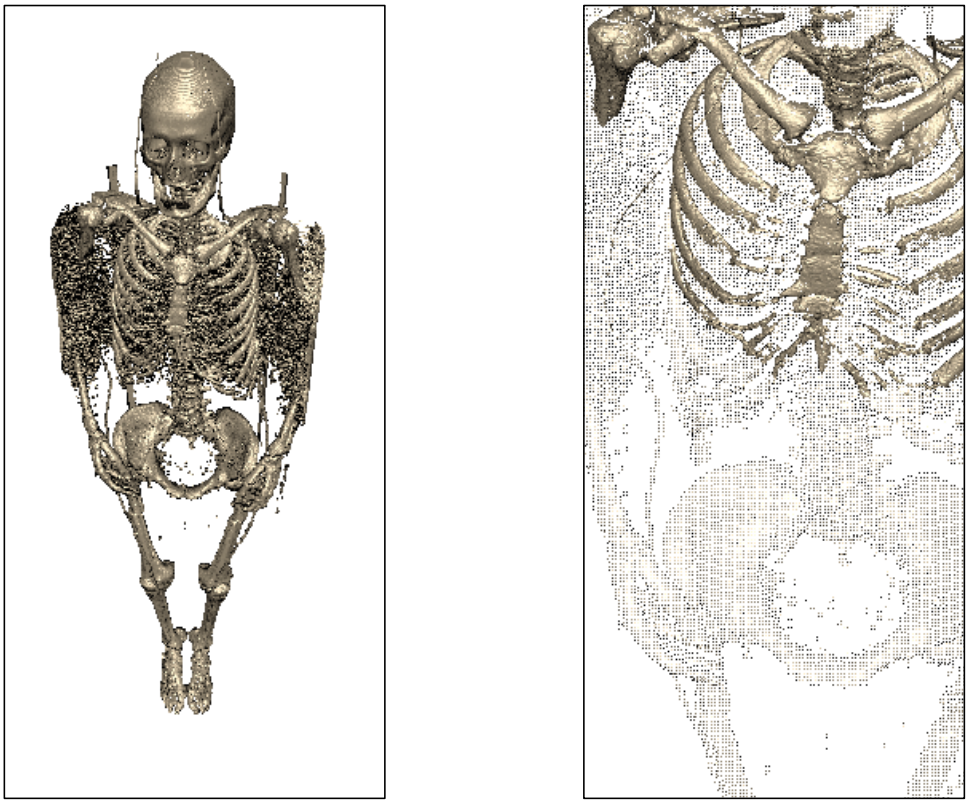

*Figure 8.* Rendering points. The left image was extracted based on the current view point. The right image show a closeup of the same extracted geometry.